HW1 ME333 Winter 2025

Author: Zhengyang Kris Weng

1. Install C, create the HelloWorld.c program, and compile and run it.

Please see `HelloWorld.c` for more details

2. Explain what a pointer variable is, and how it is different from a non-pointer variable

A pointer variable in programming is a special type of variable that stores the memory address of another variable rather than directly holding a value. Non-pointer variables operate on the actual value they hold, while pointer variables operate on references to values, making them fundamental in memory management and certain programming paradigms.

3. Explain the difference between interpreted and compiled code.

Interpreted code is translated and executed line by line by an interpreter at runtime, making it more flexible and platform-independent but typically slower. In contrast, compiled code is translated entirely into machine code by a compiler before execution, resulting in faster performance but requiring separate compilation for each platform.

4. Write the following hexadecimal (base-16) numbers in eight-digit binary (base-2) and three-digit decimal (base-10). Also, for each of the eight-digit binary representations, give the value of the most significant bit. (a) 0x1E. (b) 0x32. (c) 0xFE. (d) 0xC4.

a. **Binary**: 00011110 **Decimal**: 30 **MSB**: 0 b. **Binary**: 00110010 **Decimal**: 50 **MSB**: 0 c. **Binary**: 11111110 **Decimal**: 254 **MSB**: 1 d. **Binary**: 11000100 **Decimal**: 196 **MSB**: 1

6. Assume that each byte of memory can be addressed by a 16-bit address, and every 16-bit address has a corresponding byte in memory. How many total bits of memory do you have?

2^16 = 65536 addresses
66536 * 8 = 524288 bits
There's a total of 524288 bits in the memory.

7. Let ch be of type char . (a) The assignment ch = ’k’ can be written equivalently using a number on the right side. What is that number? (b) The number for ’5’ ? (c) For ’=’ ? (d) For ’?’ ?

'k' = 107; '5' = 53; '=' = 61; '?' = 63

8. What is the range of values for an unsigned char , short , and double data type?

unsigned char: 0 to 255;
short: -2^15 to 2^15-1 = -32768 to 32767;
double: -1.8 * 10^308 to 1.8 * 10^308 per IEEE 754; （I know Kevin says +/-2^2048 in the video but I'm not convinced)

10. Explain the difference between unsigned and signed integers.

Unsigned integers use all the available bits to represent only non-negative numbers, allowing a range from 0 to 2^n−1, where n is the number of bits. Signed integers use one bit (the most significant bit) to indicate the sign - 0 for positive and 1 for negative. The remaining n−1 bits represent the value. They are typically represented using two's complement for negative values, giving a range from −2^(n−1) to 2^(n−1)−1.

11. (a) For integer math, give the pros and cons of using char s vs. int s. (b) For floating point math, give the pros and cons of using float s vs. double s. (c) For integer math, give the pros and cons of using char s vs. float s.

a. When using char versus int for integer math, the primary advantage of char is its smaller memory usage, as it typically takes only 1 byte compared to 4 bytes for an int, making it more memory-efficient for small-range values. However, the major downside of char is its limited range, typically from -128 to 127 for signed or 0 to 255 for unsigned, which makes it unsuitable for larger values and increases the risk of overflow.
On the other hand, int offers a much wider range (usually from -2^31 to 2^31-1 for a 32-bit integer), making it more suitable for general-purpose arithmetic without concerns about overflow. The downside of using int is that it requires more memory, which may be less efficient for small data sets or embedded systems where memory is limited.

b. When it comes to floating-point math, the primary difference between float and double is the precision and range of values they can represent. float typically uses 4 bytes (32 bits) and offers a precision of about 7 decimal digits, making it suitable for situations where memory is limited or when slightly lower precision is acceptable. The disadvantage of using float is its reduced precision, which can lead to rounding errors in computations that require higher accuracy.
On the other hand, double uses 8 bytes (64 bits) and provides about 15 decimal digits of precision, offering greater accuracy and a larger range of values. This makes double the preferred choice for most scientific and engineering calculations where precision is critical. The downside of double is its higher memory usage and computational cost, as it requires more processing power and storage. Therefore, while float is more memory-efficient, double is generally preferred when accuracy is more important than memory constraints.

c. char is an integer type, typically used to represent small integer values, with a range of -128 to 127 for signed char or 0 to 255 for unsigned char. It is highly memory-efficient, taking only 1 byte, making it a good choice for situations where memory is limited and the values fit within its range. However, char cannot represent fractional numbers and may require type promotion to int during arithmetic, which can negate some of its memory advantages.
On the other hand, float is a floating-point type used for representing real numbers (with decimals), but it is not designed for integer math. While float has a much wider range and can handle fractional values, it introduces potential precision issues due to its limited precision (about 7 decimal digits). float also takes 4 bytes of memory, which is significantly more than char. Using float for integer math can lead to unnecessary complexity and inefficiency since it's primarily intended for continuous values and introduces rounding errors for integers. Therefore, for pure integer math, char is more appropriate.

16. Technically the data type of a pointer to a double is "pointer to type double ." Of the common integer and floating point data types discussed in this chapter, which is the most similar to this pointer type? Assume pointers occupy eight bytes.

A pointer to a double is most similar to a pointer to a long int because both types typically occupy the same amount of memory and have similar alignment requirements on most systems. On a 64-bit architecture, pointers generally occupy 8 bytes, regardless of the data type they point to. However, the similarity lies in the size of the data they reference; both double and long int typically require 8 bytes of memory.

Although double and long int differ in their usage—one being for floating-point operations and the other for integers—their shared size and alignment properties make their pointer types comparable.

17. The addresses have the following hexadecimal contents

(a) Initial state:

```
0xB0..0xB3 (i): unknown
0xB4..0xB7 (j): unknown
0xB8 (kp): unknown
0xB9 (np): unknown
```

(b) After kp = &i;:

```
0xB0..0xB3 (i): unknown
0xB4..0xB7 (j): unknown
0xB8 (kp): 0xB0 00
0xB9 (np): unknown
```

(c) After j = *kp;:

```
0xB0..0xB3 (i): unknown
0xB4..0xB7 (j): unknown
0xB8 (kp): 0xB0 00
0xB9 (np): unknown
```

(d) After i = 0xAE;:

```
0xB0..0xB3 (i): 0xAE 00 00 00
0xB4..0xB7 (j): unknown
0xB8 (kp): 0xB0 00
0xB9 (np): unknown
```

(e) After np = kp;:

```
0xB0..0xB3 (i): 0xAE 00 00 00
0xB4..0xB7 (j): unknown
```

```
0xB8 (kp): 0xB0 00
0xB9 (np): 0xB0 00
```

(f) After *np = 0x12;:

```
0xB0..0xB3 (i): 0x12 00 00 00
0xB4..0xB7 (j): unknown
0xB8 (kp): 0xB0 00
0xB9 (np): 0xB0 00
```

(g) After j = *kp;:

```
0xB0..0xB3 (i): 0x12 00 00 00
0xB4..0xB7 (j): 0x12 00 00 00
0xB8 (kp): 0xB0 00
0xB9 (np): 0xB0 00
```